Example for: Representing and understanding the carbon cycle using the theory of compartmental dynamical systems

Carlos A. Sierra, Verónika Ceballos-Núñez, Holger Metzler, Markus Müller

4/9/2018

This document presents an example with a specific model for some of the ideas expressed in the manuscript. It contains R code to reproduce all results. It consists of two parts: first, a description of the model used in the example; second, a explicit computation of the two cases represented in Figure 1, the autonomous and the non-autonomous case.

The 8-pool linear model

Weng & Luo (2011) proposed an eight-pool model to simulate the observed C dynamics at Duke forest, North Carolina. The model has 8 pools (foliage, woody biomass, fine roots, metabolic litter, structural litter, fast SOM, slow SOM, and passive SOM), and can be expressed in the general form of the linear autonomous model of Equation 1. Using inverse parameter estimation, Weng & Luo (2011) found the following parameterization for the model's components

```
# Inputs by photosynthesis (GPP) in g C m-2 day-1
U=3.370

#Partitioning of photosynthetic producs (unitless)
b=c(0.14,0.26,0.14,rep(0,5))

#Diagonal matrix with cycling rates for each pool (day-1)
C=diag(c(0.00258,0.0000586,0.002390,0.0109,0.00095,0.0105,0.0000995,0.0000115))

#Values of transfer coefficients among pools (unitless)
f41=0.9; f43=0.20
f51=0.1; f52=1; f53=0.80
f64=0.45; f65=0.275; f67=0.42; f68=0.45
f75=0.275; f76=0.296
f86=0.004;f87=0.01

#Matrix of transfers coefficients (unitless)
A=matrix(c(-1, 0, 0, 0, 0, 0, 0, 0,
           0, -1, 0, 0, 0, 0, 0, 0,
           0, 0, -1, 0, 0, 0, 0, 0,
           f41, 0, f43, -1, 0, 0, 0, 0,
           f51, f52, f53, 0, -1, 0, 0, 0,
           0, 0, 0, f64, f65, -1, f67, f68,
           0, 0, 0, 0, f75, f76, -1, 0,
           0, 0, 0, 0, 0, f86, f87, -1),
         byrow=TRUE,nrow=8,ncol=8)

# Initial conditions (g C m-2)
X0=c(250,4145,192,93,545,146,1585,300)
```

The compartmental system

This linear autonomous system has the following compartmental matrix $\mathbf{B} = \mathbf{AC}$

```
# The autnomous compartmental matrix
B=A%*%C
B
```

```
##           [,1]      [,2]      [,3]      [,4]        [,5]      [,6]
## [1,] -0.002580  0.00e+00  0.000000  0.000000  0.00000000  0.000000
## [2,]  0.000000 -5.86e-05  0.000000  0.000000  0.00000000  0.000000
## [3,]  0.000000  0.00e+00 -0.002390  0.000000  0.00000000  0.000000
## [4,]  0.002322  0.00e+00  0.000478 -0.010900  0.00000000  0.000000
## [5,]  0.000258  5.86e-05  0.001912  0.000000 -0.00095000  0.000000
## [6,]  0.000000  0.00e+00  0.000000  0.004905  0.00026125 -0.010500
## [7,]  0.000000  0.00e+00  0.000000  0.000000  0.00026125  0.003108
## [8,]  0.000000  0.00e+00  0.000000  0.000000  0.00000000  0.000042
##            [,7]       [,8]
## [1,]  0.000e+00  0.000e+00
## [2,]  0.000e+00  0.000e+00
## [3,]  0.000e+00  0.000e+00
## [4,]  0.000e+00  0.000e+00
## [5,]  0.000e+00  0.000e+00
## [6,]  4.179e-05  5.175e-06
## [7,] -9.950e-05  0.000e+00
## [8,]  9.950e-07 -1.150e-05
```

A compartmental matrix must follow three important properties: 1) all diagonal elements must be non-possitive

```
diag(B)<=0
```

```
## [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

2. All off-diagonal elements must be non-negative

```
B[lower.tri(B)]>=0 # Lower diagonal elements
```

```
##  [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [15] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

```
B[upper.tri(B)]>=0 # Upper diagonal elements
```

```
##  [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [15] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

3. The sum of all elements from each colum must be non-possitive

```
round(colSums(B), 5) # round to 5 digits
```

```
## [1]  0.00000  0.00000  0.00000 -0.00600 -0.00043 -0.00735 -0.00006 -0.00001
```

```
round(colSums(B), 5) <= 0
```

```
## [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

This system has a steady-state solution given by $\mathbf{x} = -\mathbf{B}^{-1} \cdot U \cdot \mathbf{b}$, which is a fixed point in the state space.

```
# Steady-state solution of the LA system in Mg C ha-1
x=(-1*solve(B)%*%(U*b))/100
x
```

```
##              [,1]
## [1,]   1.8286822
## [2,] 149.5221843
## [3,]   1.9740586
## [4,]   0.4761284
## [5,]  13.6928421
## [6,]   0.8111121
## [7,]  61.2883566
## [8,]   8.2650977
```

Simulations for the autonomous case

We will run now a set of 10 simulations with different values of the initial conditions to observe that in all cases the solution trajectories converge to a single point in the state-space. Since the dimension of the system is 8, we will plot the results in 2-D plots by summing all pools that belong to the vegetation pool, and all pools that belong to the soil and litter pools.

First, we will create a vector of time-steps to compute the solution of the system, and load the SoilR package (Sierra, Müller, & Trumbore, 2012) that will be used to compute the trajectories. We will run simulations for 300 years at a daily time-step. A function that creates the model using SoilR's sintax is available in the file modWengLuo.R, and we will load this file into our working environment.

```
days=seq(from=0,to=365*300)
library(SoilR)
source("modWengLuo.R")
```

Now we will create 10 random perturbations to the initial conditions using an uniform distribution with a minimum of 0.5 and a maximum of 1.5. The idea is to take these random numbers and multiply them by the fixed initial conditions. The resul is a set of 10 different initial conditions in a range between half and twice the values of the initial conditions from the original similation in Weng & Luo (2011).

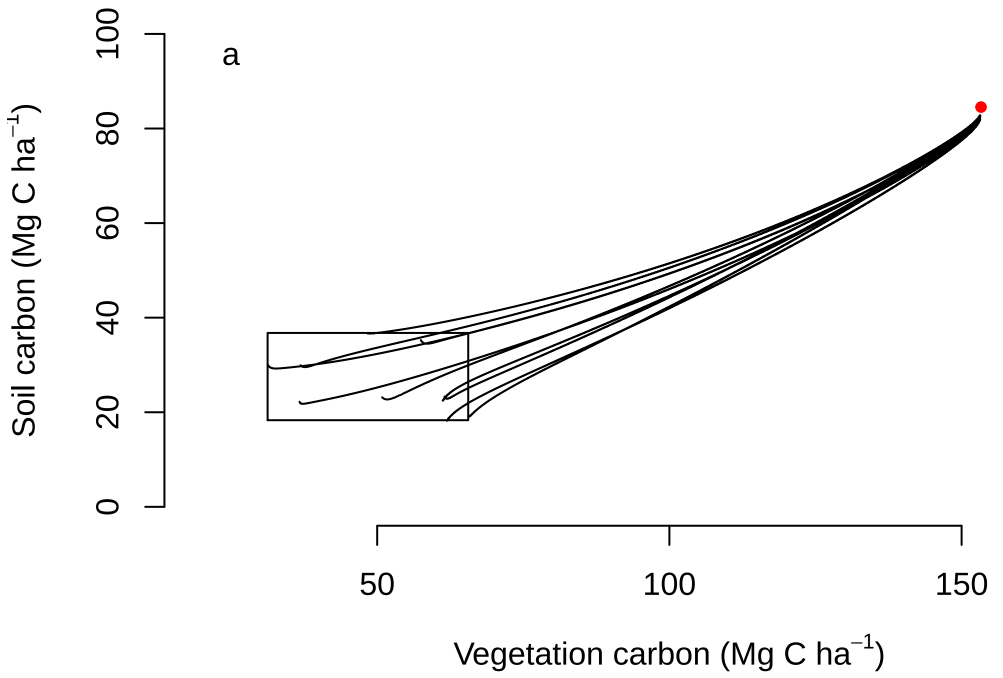
```
set.seed(21) #Fixed random number generator for reproducibility
rx0=replicate(10, runif(8, 0.5, 1.5))
```

Using a loop, we will compute 10 different simulations with 10 different initial conditions

```
n=10
vLA=NULL
sLA=NULL
for(i in 1:n){
  x0=rx0[1:8,i]*X0
  mod=modWengLuo(t=days,U=U,b=b,A=A,C=C,X0=x0,xi=1,pass=TRUE)
  ct=getC(mod)/100
  V=rowSums(ct[,1:3])
  S=rowSums(ct[,4:8])
  vLA<-cbind(vLA,V)
  sLA<-cbind(sLA,S)
}
```

We can now plot all solution trajectories for the aggregated vegetation and soil pools. The range of initial conditions will be enclosed inside a polygon.

```
matplot(vLA, sLA, type="l", xlim=c(20,180), ylim=c(0,100),
        xlab=expression(paste("Vegetation carbon (Mg C ", ha^{-1},")")),
        ylab=expression(paste("Soil carbon (Mg C ", ha^{-1}, ")")), lty=1, col=1, bty="n")
polygon(x=rep(range(vLA[1,]), each=2), y=c(range(sLA[1,]), rev(range(sLA[1,]))))
points(sum(x[1:3]), sum(x[4:8]), pch=20, col=2)
legend("topleft", "a", bty="n")
```

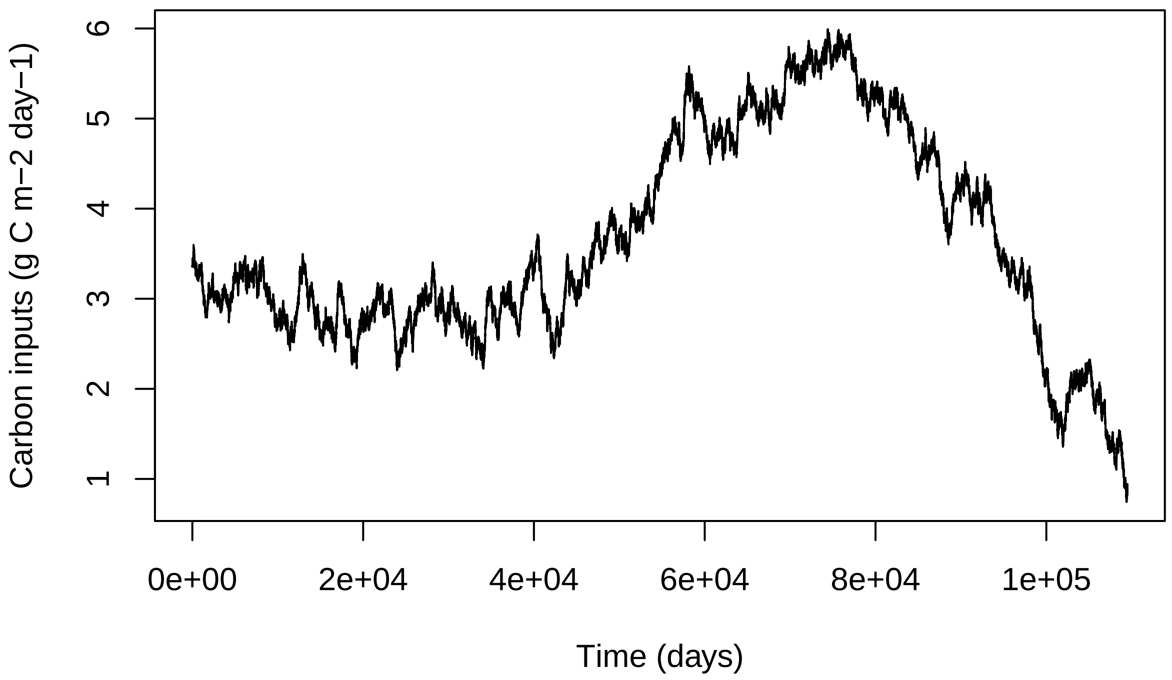


Notice that all trajectories converge to the fixed point (steady-state) given by the vector **x** that we previously computed and plotted as a red dot.

Simulations for the non-autonomous case

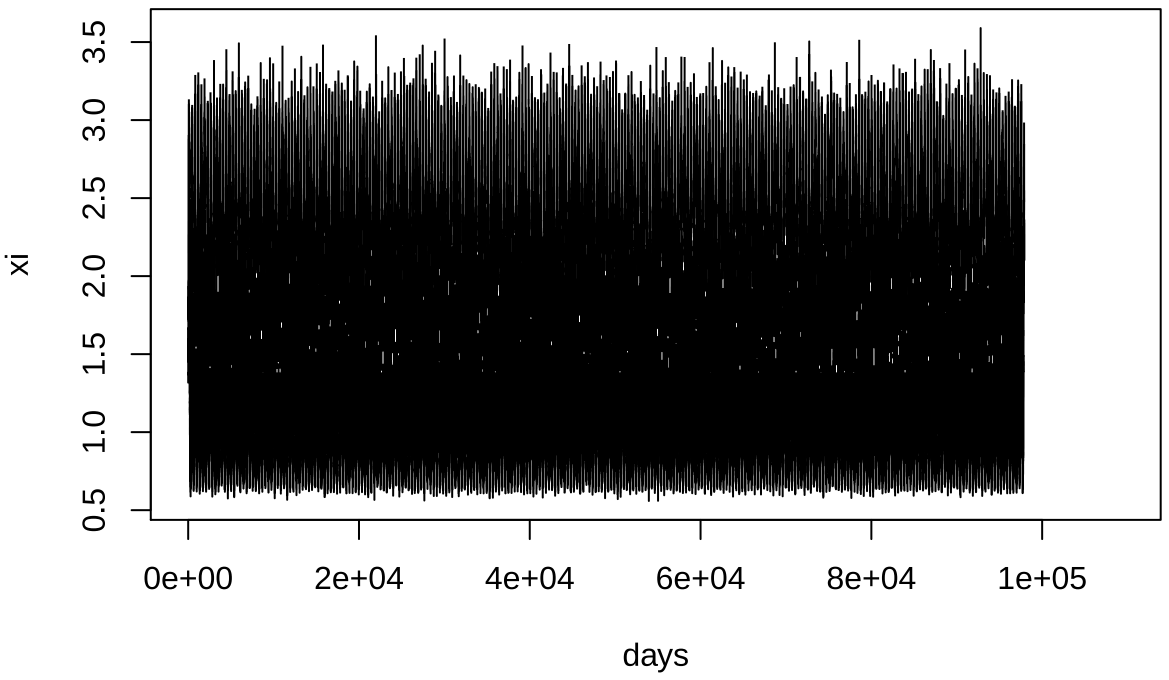
We will create now random photosynthetic inputs by creating a random walk, with an initial value equivalent to U plus random noise from a normal distribution with mean 0 and standard deviation of 0.01.

```
set.seed(19)
rwI=U+cumsum(rnorm(days,0,0.01)) #Randow walk
rI=data.frame(time=days,In=rwI)
plot(rI,type="l", xlab="Time (days)", ylab="Carbon inputs (g C m-2 day-1)" )
```



We will also create a seasonal cycle of temperatures with added random noise in order to perturb all cycling rates of the matrix **B** with a rate modifier function $\xi(t)$. In this case we use a Q10 function (Sierra et al., 2012) to obtain the rate modifier from artificial temperature data with a mean of 15 degrees Celcius, a seasonal amplitude of 10 degrees C, and daily random noise from a normal distribution with a mean of 0 and standard deviation of 1 degree C.

```
xi=data.frame(days,xi=fT.Q10(Temp=15+10*sin(2*pi*days/365)+rnorm(n=length(days),mean=0,sd=1)))
plot(xi, type="l")
```

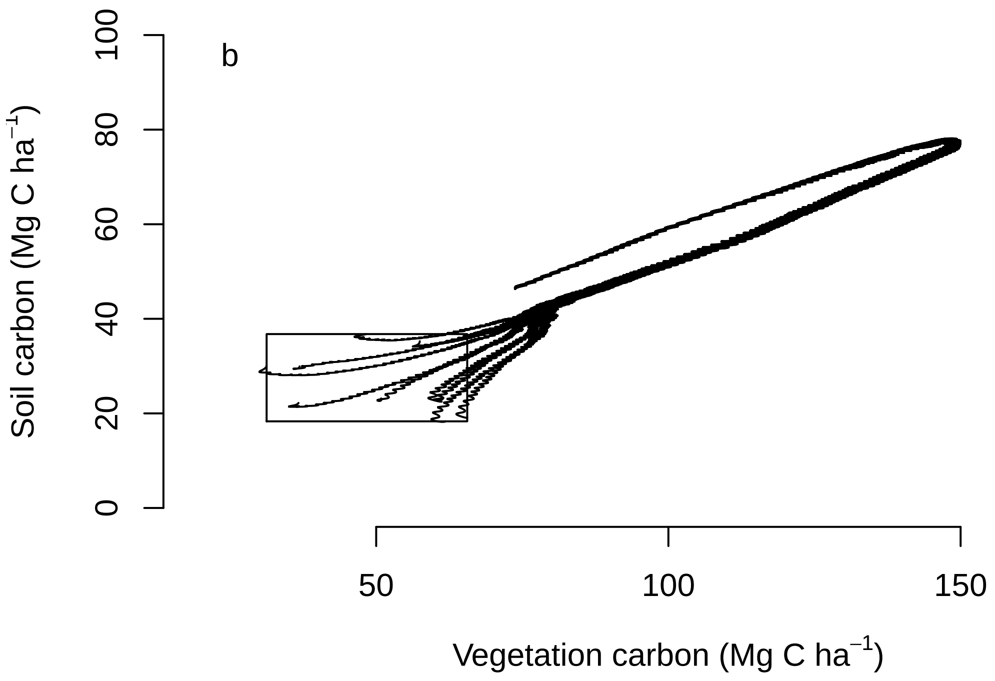


Now we run a set of 10 simulations with different initial conditions as in the first set of simulations, but using now the perturbing random signals for the inputs and the cycling rates.

```r
n=10
vLNA=NULL
sLNA=NULL
for(i in 1:n){
  x0=rx0[,i]*X0
  mod=modWengLuo(t=days,U=rI,b=b,A=A,C=C,X0=x0,xi=xi,pass=TRUE)
  ct=getC(mod)/100
  V=rowSums(ct[,1:3])
  S=rowSums(ct[,4:8])
  vLNA<-cbind(vLNA,V)
  sLNA<-cbind(sLNA,S)
}
```

We can plot now the results,

```r
matplot(vLNA, sLNA, xlab=expression(paste("Vegetation carbon (Mg C ", ha^{-1},")")),
        ylab=expression(paste("Soil carbon (Mg C ", ha^{-1}, ")")), col=1, lty=1,
        xlim=c(20,180), ylim=c(0,100), type="l", bty="n")
polygon(x=rep(range(vLNA[1,]), each=2), y=c(range(sLNA[1,]), rev(range(sLNA[1,]))))
legend("topleft", "b", bty="n")
```



```r
par(mfrow=c(1,1))
```

Notice that in all cases, the trajectories start at different values of the state space, increase after certain value and subsequently decline. They converge among each other but do not reach any fixed point (steady state). The overall behavior is mostly controlled by the dynamic behavior of the photosynthetic inputs U, which for this case increase to a certain level and then decline, causing the increase and decrease of vegetation and soil carbon observed in these simulations.

With this example, we wanted to show that the concepts of fixed point or steady-state are irrelevant for non-autonomous simulations. Furthermore, the concept of attractor in the non-autonomous case applies at the level of trajectories or sets in the state space, but not at the level of fixed points.

References

Sierra, C. A., Müller, M., & Trumbore, S. E. (2012). Models of soil organic matter decomposition: The SoilR package, version 1.0. *Geosci. Model Dev.*, *5*(4), 1045–1060.

Weng, E., & Luo, Y. (2011). Relative information contributions of model vs. Data to short- and long-term forecasts of forest carbon dynamics. *Ecological Applications*, *21*(5), 1490–1505. https://doi.org/10.1890/09-1394.1